Introduction

The enormous development in information technology has persuaded most enterprises to integrate applications among its heterogeneous systems. Heterogeneous systems are developed using several platforms, computer languages, and different technologies. They exist from not only within the enterprise but also within its business partner systems. A traditional custom integration solution can cause inconsistent and production chaos; hence, a modern systematic integration approach is required for improving the efficiency and providing less maintenance. There are several approaches such as Enterprise Application Integration (EAI), Message-Oriented Middleware (MOM), and Web Services (WS) that are used for integrating heterogeneous systems.

Among the mentioned approaches, WS is the most promising technology to reach the interoperation of heterogeneous environments. It uses standard specifications such as SOAP and WSDL that are XML based. However, traditional centralized Web service is synchronous and normally used in the way that all service clients send requests and get responses from a service provider. So it cannot support certain situations because service clients must have updated data immediately from the service provider. For this reason, the service clients are forced to check regularly for new or updated data that increases network traffic and service response time. Also, the workload on a service provider can be very high when it receives numerous requests from many service clients simultaneously.

The Publish/Subscribe interaction scheme is receiving increasing attention and is claimed to provide the loosely coupled form of interaction required for large scale settings. Publish/Subscribe is a broad technology domain and consists of many solutions for different environments. The evolution of Publish/Subscribe has followed two main objectives, namely an increased decentralization and an increased orientation on the participants’ specific needs. Experiences in building Publish/Subscribe solutions and implementing them suggest that no single solution is able to meet the demands of differing application environments and their requirements. This is evident in the number of Publish/Subscribe related standards, implementations, protocols, and algorithms. On the web front, Publish/Subscribe enables asynchronous communication of various web components such as web pages and web sites.
Industry standards have defined the conventions, interfaces, and application programming interfaces (APIs) for creating interoperable Publish/Subscribe based products and solutions that use the technology. Publish/Subscribe infrastructure is responsible for matching events to related subscriptions and delivering matching events to interested consumers. Having understood the foundations of Publish/Subscribe architecture and its tremendous business potential, HTC has developed Publish/Subscribe system for a leading finance company to ease their travel management business process needs. The designed Publish/Subscribe system decoupled the data flow primarily due to Publish/Subscribe framework design and created a change in the work culture. The Publish/Subscribe framework is extensively deployed in stock and portfolio systems for ease of management.

“Information overload has forced people to selectively subscribe for events/messages. Effectively build your services to attract more subscribers”.

- Practice Head - Big Data, HTC Global Services Inc.

What are Notifications?

Notifications are critical entities of the Publish/Subscribe system. It is the act of transmitting notification message to an interested party which is a critical element of decentralized communication. A Publisher detects an event and then publishes it in the form of a notification. A notification encapsulates information pertaining to the observed event. The notification can also be called as the event message. After the notification has been published, it is the duty of the Publish/Subscribe system to deliver the message to interested recipients – the Subscriber. This is built around standard interaction schemes or frameworks and has opened up the doors for variety of novel business applications. Several tools (for example, Kafka) are built on the concept of Publish/Subscribe framework. More information on Kafka is available in Apache Kafka-Your Event Stream Processing Solution white paper.

Who is the Notification Producer?

When the concept of notification is outlined, it is essential to understand the components of the system. A NotificationProducer is responsible for generating notifications. At a concept level, it is a web service that implements the message exchanges associated with the NotificationProducer and supports one or more Topics. The concept of Topic is extensively used in the context of Publish/
Subscribe discussions. It is capable of distributing notification messages across channels. It maintains a list of subscription resources and when it has a notification message to distribute, it matches the message against the registered subscriptions in each subscription list. If it identifies a match, a notification is issued to the NotificationConsumer associated with that subscription. A Web service that implements message exchanges associated with NotificationProducer can be a Publisher or NotificationBroker. More often, it is called as NotificationBroker. The Broker receives notification messages produced by a separate Publisher entity. This is a very simple outline of NotificationProducer which forms the foundation for most of our current social network frameworks.

**Who are Publishers?**

A Publisher is an entity that creates NotificationMessages based on situation(s) that it is capable of detecting and translating into NotificationMessage artifacts. It does not need to be a web service. Publishers serve as agents to publish events through the system by specifying values for a set of well-defined attributes. They provide information by creating streams of messages which contain a header describing application-specific information and a payload capturing the content of the message. Publishers generally are not aware of the Subscribers/Consumers.

**Who are Consumers?**

A NotificationConsumer is a web service that receives NotificationMessages from a NotificationProducer. The consumers are also called 'Subscribers'. Subscribers are usually interested in particular events or event patterns. They have the ability to express their interest in an event, or a pattern of events, and are subsequently notified of any event, generated by Publisher that matches their registered interest. They register their data interests with a MessageBroker in a subscription language that the Broker supports. This is achieved through appropriate subscriptions and they wait until they are informed about a matching event. Subscribers register their interest in events by typically calling a subscribe operation on the event service without knowing the effective sources of these events. This subscription information remains stored in the event service and is not forwarded to Publishers. Having outlined the producers and subscribers, it is essential to understand their interaction mechanisms and the kind of mediators required to achieve this. Industry has nick named the mediator as an event
What is an Event Notification Service?

An Event Notification Service is a facility that connects mutually anonymous parties. The service acquires, filters, and delivers information about events. A notification engine builds and maintains an index structure of the subscriptions and uses the index table to forward notifications to Subscribers. The engine offers necessary interfaces for Subscribers and Publishers that allow them to subscribe, unsubscribe, and publish content. Publish/Subscribe interaction scheme relies on an event notification service, providing storage and management for subscriptions and efficient delivery of events. Such an event service represents a neutral mediator between Publishers, acting as producers of events, and Subscribers acting as consumers of events. The ability to be aware of rich stream of events with minimal exertion and immediately detect critical events for further processing is central to any successful person or organization.

The event notification service decouples the Publisher and the Subscribers. The event message is transferred to event service and then it is transferred to the Subscribers. Though at the block level a service is essential to connect the Publisher and Subscriber entities, it is essential to understand the core block of the service.

What is an Event Service Broker?

Event Service Broker is an entity that registers events from producers acting as temporary store. It delivers events to Subscribers. Inside the Broker, arriving subscriptions are stored as continuous queries that will be applied to all incoming message queries, which remain effective until they are explicitly deleted. Incoming messages are processed on-the-fly against all stored queries. For each message, the Broker determines the set of queries matched by the message. A query result is created for each matched query and delivered to its Subscriber in a timely fashion. The event service propagates the event to all relevant Subscribers; it can thus be viewed as a proxy for the Subscribers. Note that every Subscriber will receive an event for every event conforming to its interest. The actual delivery depends on the Publish/Subscribe solution being used. In other words, one could fine tune the Publishers and Subscribers architecture and the way they interact with each other through the Broker.
**Publish/Subscribe Model**

**Pull-Based Architecture**

Majority of Publish / Subscribe model usages are in pull-based architecture. When there is a new or updated data message available at a Publisher, the Publisher sends a notification message to a Broker. The Broker will then circulate that notification message to registered Subscribers. To get the data message, the Subscribers must send request messages to the Publisher. For maintaining the status of transmissions, acknowledgements must be sent by the Subscribers to the Brokers. The figure describes the interaction between the Publisher and Subscriber.
What is Publish/Subscribe Framework?

The Publish/Subscribe interaction pattern provides subscribers with the ability to express their interest in an event or pattern of events, to be notified subsequently of any event, generated by a publisher that matches their registered interest. In other terms, producers publish information on a software bus (an event manager) and consumers subscribe to the information they want to receive from that bus. Decoupling the production and consumption of information increases scalability by removing all explicit dependencies between interacting participants. Removing these dependencies strongly reduces coordination and thus synchronization between different entities, and makes the resulting communication infrastructure well adapted to distributed environments. For example, notification based interaction is widely used to ensure consistency of
web caches widely deployed across the world when they download web contents. When there is any change in the web server, it is notified to the web proxies. This implements a limited form of Publish/Subscribe interaction in which web proxies act as Subscribers and the web server as the Publisher.

HTC has been experimenting on Publish/Subscribe framework and mapping various business cases leading to increased return on investment (ROI). HTC has mapped Publish/Subscribe solution for various clients. Using the solutions, customers subscribed for specific topics which were published by the Publishers. When event/messages are initiated in the forum or on the subscribed topic, the customers are notified. The customers participate in the discussions and become a producer to generate messages.

**What is Message Oriented Communication?**

Queuing is considered as the heart of Publish / Subscribe solutions. It is a basic solution in achieving reliability in data communications. Queuing also supports disconnections during which the message cannot be sent. Message queuing is the basic ingredient for achieving decoupled communications. Publish/Subscribe is based on message queuing and message oriented middleware. Message queuing is a communication method that employs message passing between a sender and a receiver with the help of a sender side message queue. A message being sent is first stored in the local message queue. After the delivery has been made, the message can be removed from the queue.

One distinction between message queuing systems and Publish / Subscribe is that they typically offer one-to-one communication and require that the receivers be explicitly defined. Publish/Subscribe on the other hand supports one-to-many and many-to-many communications and the Subscribers can be defined implicitly by the event message being delivered and priority subscriptions that subscribers have set.

**What are Topic-based Systems?**

Topic-based systems are similar to newsgroups. The users show their interest by registering in a group (a topic). All messages related to that topic are broadcasted to all users participating in that specific group. In topic-based systems, a message belongs to one of a fixed set of what are variously referred to as groups, channels, or subject. Subscription targets a group, channel, or
subject, and the user receives all events that are associated with that group. Brokering a connection between Publishers and Subscribers is the act of connecting channel supplier with channel consumer.

What are Content-based systems?

With Content-based Publish/Subscribe, Subscribers have the added flexibility of choosing the filtering criteria along as many dimensions as event attributes, without requiring pre-definition of subject. When an event arrives at a router, its content must be matched against the set of all registered subscriptions to find interested subscriptions, an operation termed as content-match.

Content-based systems are not constrained to the notion that a message must belong to a particular group. Instead, the decision of to whom the message must be directed is made on a message-by-message basis, established on a query or predicate issued by a Subscriber. The advantage of a content-based system is its flexibility. It provides the Subscriber just the information they need. The Subscriber need not have to learn a set of topic names and their content before subscribing. A Publish/Subscribe system must handle information dissemination across distinct authoritative domains, heterogeneous platforms, and a large dynamic population of Publishers and Subscribers. Upon receiving information from Publishers, the Publish/Subscribe brokers then filter and deliver information to Subscribers whose interests match the topic and content of the information.

“Information overload has forced people to selectively subscribe for events/messages. Effectively build your services to attract more subscribers”.

- Practice Head - Big Data, HTC Global Services Inc.
Publish/Subscribe Systems and DB Systems

A Publish/Subscribe system maintains a database of subscriptions, where each subscription is a Boolean expression. For example, each subscription in the Publish/Subscribe system of a diverse online vendor may describe the conditions under which a customer may purchase a product. A customer interested in acquiring a camera may post their requirement as a subscription to the vendor’s Publish/Subscribe system by providing the Boolean expression.

When an event occurs, the Publish/Subscribe system reports all subscriptions in its database that are matched (or satisfied by the event). Customers who posted these matching subscriptions will be notified. Publish/Subscribe systems are used in diverse applications with varied performance requirements. For example, in some applications events occur at much higher rate than the posting/removal of subscriptions, while in other applications the subscription rate may be much higher than the event rate, and in yet other applications the two rates may be comparable. Optimal performance in each of these scenarios may result from deploying a different data structure for subscriptions or a different tuning of the same structure. Many commercial applications of Publish/Subscribe systems have thousands of attributes and millions of subscriptions. So, scalability in terms of number of attributes and number of subscriptions is critical.

Publish/Subscribe and event-based systems are very different from database systems because they enable data dissemination from publishers to subscribers in the present and future. This contrasts the traditional database model in which queries are performed on existing data that is available in a database. The notions of database query and subscription are similar, but the query is about the past whereas the subscription is about the future when it is issued.

Conclusion

Publish/Subscribe systems are revolutionizing the industrial event stream processing solutions. These solutions help the business world to simplify their data pipeline systems and build data ingestion capabilities. The solutions work well with various complex event stream processing solutions. HTC has established its presence in stream processing and Publish/Subscribe solutions and has been helping customers stabilize their business streams.
Acronyms

The acronyms used in this white paper and their expansion are provided below:

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Expansion</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAI</td>
<td>Enterprise Application Integration</td>
</tr>
<tr>
<td>MOM</td>
<td>Message-Oriented Middleware</td>
</tr>
<tr>
<td>ROI</td>
<td>Return on Investment</td>
</tr>
<tr>
<td>WS</td>
<td>Web Services</td>
</tr>
</tbody>
</table>

References:

[1] Publish / Subscribe Architecture with Web Services  
http://www.ipedr.com/vol9/6-I00008.pdf

[2] Publish-Subscribe Notification for Web Services  

[3] Publish-Subscribe based Communication Model  
http://ethesis.nitrkl.ac.in/6074/1/E-198.pdf

http://members.unine.ch/pascal.felber/publications/CS-03.pdf

https://books.google.co.in/books?isbn=1118354281

http://www.cise.ufl.edu/~sahni/papers/pubsubISCC.pdf
About HTC's Big Data CoE

HTC’s Center of Excellence for Big Data Management and Analytics brings in mature technologies and thought leadership. Our dedicated R&D team develops highly customized and cost-effective cutting edge solutions to enable clients manage and understand big data for improved and quicker decision making.

This white paper was developed by HTC's Big Data CoE.